OPIC
OFFICE DE LA PROPRIÉTÉ INTELLECTUELLE DU CANADA

CIPO
CANADIAN INTELLECTUAL PROPERTY OFFICE

(12) (19) (CA) **Demande-Application**

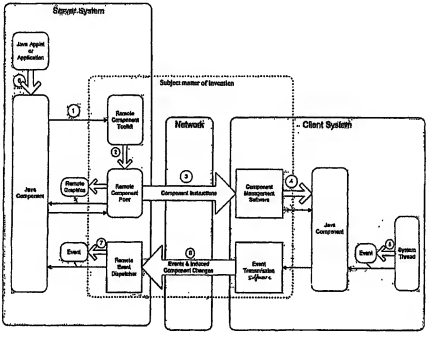
(21) (A1) **2,207,746**
(22) 1997/06/13
(43) 1998/12/13

(72) SIKS, Andrew R., CA
(71) Ironside Technologies Inc., CA

(51) Int.Cl.[6] G06F 9/45, H04L 12/16, H04L 29/02

(54) **METHODE DE MANIPULATION DE COMPOSANTES LOGICIELLES PAR RESEAU, PERMETTANT D'OBTENIR UNE MEILLEURE PERFORMANCE ET DE REDUIRE LE TRAFIC RESEAU**

(54) **METHOD OF MANIPULATING SOFTWARE COMPONENTS THROUGH A NETWORK WITH ENHANCED PERFORMANCE AND REDUCED NETWORK TRAFFIC**




(57) Méthode de manipulation de composantes d'application au moyen d'un réseau permettant d'obtenir un rendement supérieur, de réduire le trafic réseau et d'offrir une application mandataire sur un ordinateur client et un système de fenêtrage à distance sur un

(57) A method of manipulating components through an network with enhanced performance and reduced network traffic includes providing a proxy application on a client computer and a remote windowing system on a server computer. The proxy application emulates, on the

OPIC
OFFICE DE LA PROPRIÉTÉ INTELLECTUELLE DU CANADA

CIPO
CANADIAN INTELLECTUAL PROPERTY OFFICE

(21) (A1) 2,207,746
(22) 1997/06/13
(43) 1998/12/13

ordinateur serveur. L'application mandataire émule, en fonction de codes d'instruction reçus d'un ordinateur serveur, les composantes d'une application exécutée par l'ordinateur serveur. Le système de fenêtrage à distance émule, à partir de paquets de données sur l'activité provenant de l'ordinateur client, l'enregistrement de données et les événements déclenchés par l'utilisateur fournis par le système de fenêtrage de l'ordinateur client. En pratique, lorsqu'un événement déclenché par l'utilisateur est transmis à l'application mandataire par le système de fenêtrage de l'ordinateur client, l'application mandataire code les données décrivant l'événement et les transmet à l'ordinateur serveur sous la forme de paquets de données sur l'activité. Sur réception d'un paquet de données sur l'activité, le système de fenêtrage à distance de l'ordinateur serveur décode ces données et les transmet à une composante sélectionnée de l'application aux fins de traitement. Les changements aux composantes résultant de l'exécution de l'événement par l'application sont codés par le système de fenêtrage à distance et transmis sous forme de codes d'instruction au système client. Sur réception des codes d'instruction, l'application mandataire exécute les changements sur l'ordinateur client. Ainsi, l'utilisateur d'un ordinateur olient peut se servir de l'application comme si elle était exécutée localement sur l'ordinateur client plutôt que sur un ordinateur serveur distant. Le trafic réseau entre l'ordinateur client et l'ordinateur serveur se limite ainsi à la communication de paquets de données codées concernant les activités et de codes d'instruction, qui sont petits et peuvent être transmis rapidement.

basis of instruction codes received from the server computer, the components of an application running on the server computer. The remote windowing system emulates, on the basis of activity packets received from the client computer, data input and user-initiated events provided by the windowing system of the client computer. In operation, when a user-initiated event is passed to the proxy application by the windowing system of the client computer, the proxy application encodes event data indicative of the event, and transmits the encoded event data to the server computer as an activity packet. Upon receipt of an activity packet, the remote windowing system in the server computer decodes the event data, and passes the event data to a selected component of the application for processing. Component changes resulting from processing of the event by the application are encoded by the remote windowing system and transmitted as instruction codes to the client system. Upon receipt of the instruction codes, the proxy application renders the component changes on the client computer. As a result, a user of the client computer is able to utilize the application as if it were running locally on the client computer, instead of running on a remote server computer. Network traffic between the client and server computers is reduced to encoded activity packets and instruction codes, which are small and can be transmitted quickly.

ABSTRACT OF THE DISCLOSURE

A method of manipulating components through an network with enhanced performance and reduced network traffic includes providing a proxy application on a client computer and a remote windowing system on a server computer. The proxy application emulates, on the basis of instruction codes received from the server computer, the components of an application running on the server computer. The remote windowing system emulates, on the basis of activity packets received from the client computer, data input and user-initiated events provided by the windowing system of the client computer. In operation, when a user-initiated event is passed to the proxy application by the windowing system of the client computer, the proxy application encodes event data indicative of the event, and transmits the encoded event data to the server computer as an activity packet. Upon receipt of an activity packet, the remote windowing system in the server computer decodes the event data, and passes the event data to a selected component of the application for processing. Component changes resulting from processing of the event by the application are encoded by the remote windowing system and transmitted as instruction codes to the client system. Upon receipt of the instruction codes, the proxy application renders the component changes on the client computer. As a result, a user of the client computer is able to utilize the application as if it were running locally on the client computer, instead of running on a remote server computer. Network traffic between the client and server computers is reduced to encoded activity packets and instruction codes, which are small and can be transmitted quickly.

## Description of the Invention

The present invention provides a process whereby a Java program running on one computer (the Server System) creates and manipulates Java Components on one or more remote computers (Client Systems), through a network, and receives Events from these Components. The Java Components on each Client System run independently of Java Components on other Client Systems, while being remotely manipulated by the Server System. By this means, the need to transmit large numbers of Java class files to the Client System is eliminated, thereby significantly reducing delays perceived by a user of the Client System.

With reference to the attached drawing:

A Java program on the Server System creates instances of Java Components, on the Server System, in a conventional manner. When the getToolKit() method of a Java Component without a peer or parent is called, an instance of a Remote Component ToolKit (RCT) is returned.

When the addNotify() method of any non-abstract Component is called, the appropriate "create" method of the RCT is called{1}. The create method instantiates a Remote Component Peer (RCP) of the correct type {2} and returns it to the Component.

The RCP communicates with Component Management Software (CMS) on the Client System {3}. The RCP transmits component-related instructions to the CMS. The CMS converts these instructions into Component method calls that result in the creation {4} and manipulation of Components on the Client System.

When an Event is generated {5} on the Client System and delivered to the postEvent() method of a Component, the Event, if significant, is codified and transmitted {6} to a Remote Event Dispatcher on the Server System. The Dispatcher re-creates the Event {7} on the Server System and calls the postEvent() method of the corresponding Component on the Server System.

CLAIMS:

1. In a network comprising at least one server computer and at least one client computer connected via a communications media, the client computer having a conventional windowing system capable of rendering components of an application on a monitor of the client computer, and further capable of passing user-initiated events to components of an application running on the client computer, a method of manipulating components of an application running on the server computer, the method comprising the steps of:

(a) monitoring the windowing system of the client computer, and, upon receipt of a user initiated event:

[i] encoding data indicative of the user initiated event to create an activity packet, the amount of data of the activity packet typically being smaller than that of the data indicative of the event;

[ii] transmitting the activity packet to the server computer;

[iii] receiving the activity packet in the server computer;

[iv] decoding the activity packet to obtain event data indicative of the user initiated event;

[v] passing the event data to the application running on the server computer to process the event;

(b) monitoring the application running on the server computer, and, upon receipt of component change data indicative of changes in a component:

[i] transmitting the component change data to the client computer;

[ii] receiving the component change data in the client computer;

[iii] passing the component change data to components running on the client computer for rendering on the monitor of the client computer.

2. The method of claim 1, wherein steps (a) and (b) of claim 1 are conducted in sequence as an event/response cycle initiated by receipt of a user initiated event at step (a) of claim 1.

3. The method of claim 1, wherein step (a) of claim 1 includes the step of filtering the user-initiated event to determine whether or not the user-initiated event is significant, steps a[i] through b[iii] of claim 1 being executed only when it is determined that the user-initiated event is in fact significant.

4. The method of claim 1, wherein step (a) of claim 1 further includes the step of storing user-induced component changes, and, prior to step a[i] of claim 1:

(1) checking components for user-induced changes;

(2) when user induced changes are found, encoding the user induced changes to create an activity packet concerning the user induced changes;

(3) transmitting the activity packet to the server computer;

(4) receiving the activity packet in the server computer;

(5) decoding the activity packet to obtain the user-induced changes; and

(6) storing the user-induced changes in association with a selected component of the application.

5. A method of manipulating components on a remote computer via a network, the method comprising the steps of:

(a) receiving component change data concerning changes in a component from an application running on a first computer;

(b) transmitting encoded data indicative of the component change data to the remote computer;

(c) receiving the encoded data in the remote computer;

(d) decoding the encoded data in the remote computer to obtain component change data concerning desired component changes;

(e) passing the component change data to a selected component running on the remote computer to render the desired component changes.

6. The method of claim 5, wherein the encoded data comprises:

(a) a component ID indicative of a component to be changed, the component ID be used in the remote computer to select a desired component to be changed;

(b) an Op Code indicative of the type of change to be effected, the Op-Code being used in the remote computer to trigger a method of the desired component; and

(c) data serving as parameters for the method of the desired component triggered in response to the Op-Code.

7. A method of manipulating components on a remote computer comprising the steps of:

(a) receiving a user initiated event from a component running on a first computer;

(b) encoding the event to create an activity packet;

(c) transmitting the activity packet to a second computer;

(d) receiving the activity packet in the second computer;

(e) decoding the activity packet to obtain the event;

(f) passing the event to a selected component of an application running on the second computer to process the event.

8. The method of claim 7, wherein the activity packet comprises:

(a) a component ID indicative of a component affected by the user-initiated event; and

(b) an Op Code indicative of the user-initiated event.

9. In a network comprising at least one server computer and at least one client computer connected via a communications media, a method of starting an application for use by a user of a client computer, the method comprising the steps of:

(a) preliminarily storing the application, a proxy application and an enabler program on a server computer;

(b) running the enabler program on the server computer to continuously monitor the communication media for a network connection attempt initiated from a client computer;

(c) transmitting the proxy application from the server computer to a client computer via the network and initializing the proxy application on the client computer;

(d) establishing, by means of the proxy application, a network connection with the enabler program on the server computer;

(e) initializing the application on the server computer in response to establishment of a network connection with the enabler program;

(f) creating, on the server computer, one or more components in accordance with start-up logic of the application;

(g) creating a respective remote component peer for each component;

(h) transmitting, by means of each remote component peer, component creation instructions to the proxy application running on the client computer;

(i) creating, on the client computer, a respective client component corresponding to each component created on the server computer, on the basis of the component creation instructions; and

(j) rendering on a monitor of the client computer, client components as required by the intended initial condition of the application so as to begin a user's experience of the application.

10. The method of claim 9, wherein the step of initializing the proxy application includes detecting characteristics of the client computer, and transmitting to the server computer data indicative of the detected characteristics.

11. The method of claim 9, wherein the step of creating, on the server computer, one or more components, includes, for each paintable component, defining and storing a respective byte array of encoded paint instructions corresponding to the paintable component.

12. The method of claim 11, wherein the step of transmitting component creation instructions to the proxy application includes, for each paintable component, the steps of:

transmitting the byte array of encoded paint instructions to the client computer; receiving the byte array in the client computer; and storing the byte array in the client computer in association with a corresponding client component.

13. The method of claim12, wherein the step of rendering client components on a monitor of the client computer comprises painting each paintable component in accordance with its respective byte array of encoded paint instructions.

14. A method of manipulating components through a network, comprising:

(a) receiving an event from a windowing system of a client computer, the event being initiated by a user of the client computer;

(b) encoding event data indicative of the user initiated event to create an activity packet;

(c) transmitting the activity packet to a server computer through the network;

(d) receiving the activity packet in the server computer;

(e) decoding the activity packet to obtain the event data indicative of the user initiated event;

(f) calling one or more methods of a component on the server computer in accordance with the event data;

(g) Receiving component change data indicative of changes in a component running on the server computer;

(h) transmitting the component change data to the client computer;

(i) receiving the component change data in the client computer; and

(j) calling one or more methods of a component running on the client computer in accordance with the component change data.

15. The method of claim 14, wherein a flush timer thread running on the server computer is used to force timely transmission of the component change data to the client computer.

16. The method of claim 14, wherein the activity packet containing encoded data indicative of a user-initiated event comprises:

(a) a component ID indicative of a component affected by the user-initiated event;

(b) an Operation Code indicating that the activity packet contains information of a user-initiated event; and

(c) an event-type code indicative of the type event initiated by the user.

17. The method of claim 14, wherein the step of transmitting component change data to the client computer comprises the steps of :

(a) transmitting an operation code indicative of the component change;

(b) if required, transmitting any data associated with the component change;

(c) transmitting an operation code for "select component'

(d) transmitting a Component ID indicative of a specific component affected by the component change; and

(e) transmitting an operation code indicative of a functional operation to be performed on the component identified by the component ID.

18. The method of claim 17, wherein the step of receiving the component change data in the client computer, comprises sequentially receiving and processing each transmission described in claim 4.

19. The method of claim 14, wherein the step of receiving a user initiated event includes determining whether or not the user-initiated event is significant; steps b-j of claim 14 being executed only when the user-initiated event is determined to be significant.

20. The method of claim 19, further comprising the step of, when the user initiated event is determined to be insignificant, storing data indicative of induced component changes related to the insignificant event.

21. The method of claim 19, further comprising, when a user initiated event is determined to be significant, the steps of:

(a) determining whether or not data indicative of induced component changes related to prior insignificant events have be previously stored, and, when it is determined that such data have been stored;

(b) encoding the data indicative of induced component changes to create an activity packet;

(c) transmitting the activity packet to the server computer through the network;

(d) receiving the activity packet in the server computer;

(e) decoding the activity packet to obtain the data indicative of the induced component changes; and

(f) storing the data indicative of the induced component changes in association with a component running on the server computer.

22. The method of claim 21, wherein the encoded data indicative of induced component changes is transmitted to the server computer prior to transmitting the event data indicative of the user initiated event.

23. The method of claim 21, wherein the activity packet containing encoded data indicative of induced component changes comprises:

(a) a component ID indicative of a component affected by the induced component change;

(b) an Operation Code indicative of a type of the induced component change; and

(c) if required, any data stored in association with the affected component as a result of the induced component change

24. In a network comprising a server computer and a client computer connected by a network media allowing two-way transmission of data between the server and client computers, and wherein each computer includes a conventional windowing system capable of facilitating interaction between components of an application running on each computer and a respective user of each computer, a method for manipulating components of an application running on the server computer in response to inputs

provided by a user of the client computer, with minimum network traffic, the method comprising:

(a) providing a proxy application on the client computer capable of emulating, on the basis of instructions codes received from the server computer, components of the application running on the server computer, the proxy application being further capable of receiving data and user-initiated events from the windowing system of the client computer, and transmitting an activity packet indicative of the data and user-initiated events input to the server computer;

(b) providing a remote windowing system on the server computer capable emulating, on the basis of codes received from the client computer, data and user-initiated events received from the windowing system of the client computer, the remote windowing system being further capable of receiving data indicative of changes in components of the application on the server computer, and transmitting instruction codes indicative of such component changes to the client computer;

(c) whereby components of an application running on the server computer can be manipulated on the basis of data and user-initiated events provided by a user of the client computer, and network traffic is limited to transmission of activity packets indicative of such data and user-initiated events from the client computer to the server computer, and component change data indicative of component changes from the server computer to the client computer.

25. In a network comprising at least one server computer and a client computer connected for communication via a network media, a method of initializing an application on a server computer for use by a user of the client computer, the method comprising the steps of:

(a) providing and continuously running an enabler program on a server computer, the enabler program being capable of monitoring the network for, and accepting, network connection requests from the client computer;

(b) retrieving an object class definition of a proxy application into the client computer, and initializing the proxy application in the client computer;

(c) using the proxy application to establish a network connection with the enabler program running on the server computer;

(d) when the enabler program on the server to accepts a network connection from the proxy application, using the enabler program to create and initializing an instance of the application's startup logic on the server;

(e) using the application's startup logic to create components consistent with the application's intended form of operation;

(f) as each component is created, creating a respective remote component peer on the server computer;

(g) using the remote component peer to transmit component creation instructions to the proxy application on the client computer; and

(h) using the component manager to create a client component on the client computer on the basis of the component creation instructions received from the server computer, the client component corresponding to and emulating a respective component on the server computer.

26. The method of claim 25, wherein the network is a TCP/IP network, and the client computer communicates with the server computer using a browser program.

27. The method of claim 26, wherein a user of the client computer triggers initialization of the application by clicking on a link which causes the browser to retrieve, from an HTTP server, an HTML page containing an HTML applet tag which identifies the object class definition of the proxy application; the browser being responsive to the applet tag to retrieve the object class definition of the proxy application from an HTTP server and create and initialize an instance of the proxy application object in a conventional manner.

28. The method of claim 26, further comprising, when the proxy application establishes a network connection with the server computer, the step of using the proxy application to detect and transmits data indicative of functional characteristics of the client computer.

29. The method of claim 28, wherein the data indicative of functional characteristics of the client computer includes the type and version of the operating system and the resolution of a monitor screen of the client computer.

30. The method of claim 28, further comprising, when the enabler program initializes the application startup logic on the server computer, the step of using the enabler program to replace the conventional windowing toolkit of the server computer with a remote component toolkit that is compatible with the proxy application running on the client computer and the detected functional characteristics of the client computer, whereby remote component peers compatible with the proxy application and functional characteristics of the client computer are created during subsequent creation of components of the application on the server computer.

31. The method of claim 27, wherein the enabler program and application, the HTML page, and the object class definition of the proxy application are each stored of respective different server computers on the network.

32. The method of claim 27, wherein any two or more of the enabler program and application, the HTML page, and the object class definition of the proxy application are stored on the same server computer.

33. In a network comprising a server computer and a client computer connected by a network media allowing two-way transmission of data between the server and client computers, and wherein each computer includes a conventional windowing system capable of facilitating interaction between components of an application running on each computer and a respective user of each computer, a system for manipulating components of an application running on the server computer in response to inputs provided by a user of the client computer, with minimum network traffic, the system comprising:

(a) a proxy application running on the client computer, the proxy application being capable of emulating, on the basis of codes received from the server computer, components of the application running on the server computer, the proxy application being further capable of receiving data and user-initiated events from the windowing system of the client computer, and transmitting an activity packet indicative of data and user-initiated events to the server computer;

(b) a remote windowing system running on the server computer, the remote windowing system being capable emulating, on the basis of codes received from the client computer, data and user-initiated events received from the windowing system of the client computer, the remote windowing system being

further capable of receiving data indicative of changes in components of the application on the server computer, and transmitting instructions codes indicative of such component changes to the client computer.

34. A system as claimed in claim 33, wherein the proxy application comprises:

(a) at least one client component capable of interacting with the windowing system of the client computer in a conventional manner to facilitate rendering of each client component and reception of data and user-initiated events;

(b) a component manager capable of instantiating and manipulating each client component in response to instruction codes received from the server computer;

(c) an event handler responsive to each client component and capable of receiving event data indicative of a user initiated event received by a client component, and transmitting the event data to the server computer as an activity packet;

(d) a client component table containing component identifier and address information for each client component, the client component table providing a look-up table whereby the component manager can select a component on the basis of a component ID received from the server computer; and

(e) a component painter capable of interacting with each client component, and the windowing system on the client computer, to control the rendering of "paintable" components on a monitor of the client computer.

35. A system as claimed in claim 34, wherein the event handler comprises:

(a) an event filter responsive to each client component for determining whether or not a user-initiated event is significant;

(b) an event transmitter responsive to the event filter for encoding and transmitting to the server computer event data as an activity packet;

(c) a change transmitter responsive to the event filter for encoding and transmitting to the server computer induced component change data as an activity packet.

36. A system as claimed in claim 35, wherein the event transmitter is caused to transmit event data only when event filter determines that the user-initiated event is significant.

37. A system as claimed in claim 35, wherein the event filter stores data indicative of induced component changes when it a user-initiated event is determined to be insignificant.

38. A system as claimed in claim 37, wherein, when the event filter determines that a user-initiated event is significant, the event filter causes the change transmitter to encode and transmit induced component changes prior to causing the event transmitter to encode and transmit the event data.

39. A system as claimed in claim 33, wherein the remote windowing system comprises:

(a) a respective remote component peer corresponding to each component of an application running on the server computer, the remote component peer being capable of interacting with its corresponding component in a conventional manner to reflect changes in the component resulting from operation of the application and to pass user-initiated events to the component for processing, the remote component peer being responsive to component changes to transmit change codes indicative of the component changes to the client computer;

(b) a remote component toolkit comprising object classes for creating a remote component peer to correspond with a respective component, during creation of each component of the application;

(c) remote event dispatcher capable of receiving activity packets from the client computer, decoding each activity packet to obtain event data, and passing the event data to a selected remote component peer as an event; and

(d) a component table including component identifiers and addresses for each remote component peer, whereby the remote event dispatcher can readily select a remote component peer on the basis of the event data received from the client computer.